VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Testování obrazových dat při různém typu rozkladu

Testing of Image Data in a Different Type of Decomposition

2010 Martin Kocurek

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

7. 5. 2010 Martin Kocurek

Abstrakt: Bakalářská práce se zabývá tématem testování obrazových dat při různém typu rozkladu. Práce obsahuje teorii k vektorové kvantizaci, neboť tento proces se využívá při kompresi jednotlivých obrazových dat. Práce obsahuje navržené typy pro vektorovou kvantizaci. Testovací aplikace využívá jazyka C++ v prostředí Microsoft Visual Studio 2008. V práci naleznete navržené rozklady pro kompresi a jejich nejlepší nastavení. Práce uvádí porovnání navržených rozkladů se standardním čtvercovým rozkladem a obsahuje výsledky a poznatky z testování.

Abstract: The Bachelor thesis describes theme testing of image data in a different type of decomposition. The work includes the theory vector quantization, since this process will use in the compression of image data. The work includes a proposed types of vector quantization. Sample application to test used C++ language and Microsoft Visual Studio 2008. In work can be found individual decompositions for compressing and the best compression settings. In written work are compared the proposed decompositions with standard square decomposition and contains the results and knowledge from the testing.

Klíčová slova: Vektorová kvantizace, Typ Rozkladu, Komprese Obrazu, C++

Key-words: Vector Quantization, Type of Decomposition, Image Compression, C++

Poděkování:
Děkuji svému vedoucímu, Ing. Lukáši Hlaváčkovi, za osobní přístup, odborné vedení mé práce, cenné rady, pomoc a čas, který mi věnoval.

Seznam použitých zkratek a symbolů

EV	- Euklidovská vzdálenost
FSVQ	- Full Search Vector Quantization
LBG	- Linde-Bazo-Gray Algorithm
LUT	- LookUp Table
NN	- Nearest Neighbor problem
SQ	- Scalar Quantization
TSVQ	- Tree Structured Vector Quantization
VQ	- Vector Quantization

Obsah

1 Úvod

Cílem této bakalářské práce je testování obrazových dat při různém typu rozkladu. Pro lepší pochopení celé této problematiky, je nutné znát základní pojmy a témata, které se vztahují v dané práci.

Prvním krokem před samotnou implementací a testováním bylo nutné nastudovat teorii principu vektorové kvantizace (dále jen VQ), kterou využiji při kompresi obrázků. Tento proces je často používán ve ztrátové kompresi dat, a celá práce s testováním je založena právě na zmíněné metodě. V samostatné kapitole proto popisuji VQ a její praktické využití pro mé další postupy.

Následující část zahrnuje navrhnutí nových vzorů rozkladu obrazových dat. Využil jsem nabytých znalostí z teorie a vhodně se pokusil rozložit obrázky pro VQ. V kapitole popisuji všechny různé typy rozkladů, jakým způsobem mi tvary rozdělili obrazové data a následné poskládání do původních obrázků. Pří návrhu jsem musel brát ohled na skutečnost, aby tvary do sebe přijatelně zapadaly, protože pracuji s obrázky ve tvaru čtverce.

K otestování nových vzoru rozkladu jsem vytvořil ukázkovou aplikaci. Samotná implementace probíhala v prostředí Microsoft Visual Studio 2008 a programoval jsem v jazyce C++. V této kapitole poukazuji na kroky, jak program postupně vznikal, s kterých hlavních částí se skládá a co vše bylo potřebné vytvořit. Zdůrazňuji jen prvky, které se vztahují k mému tématu, a bez kterých by nebylo možné důkladně nové vzory testovat.

Po dokončení aplikace jsem testoval navržené rozklady pro kompresi pomocí VQ. V celé části popisuji testy jednotlivých rozkladů. Využil jsem všechny možnosti nastavení komprese a zaznamenal dosažené výsledky pro daný tvar. Na tomto základě jsem byl schopen určit nejlepší nastavení pro rozklad.

Při testování jsem nesrovnával mé nové tvary mezi sebou, důležité pro mě bylo zjistit, jestli nějaký tvar není vhodnější než standardní řešení. Proto poslední kapitolou v mé práci je porovnání jednotlivých navržených rozkladů se čtvercovým rozkladem. Využil jsem výsledků z předchozí části a zkoumal odlišnosti a shody obrázků po kompresi mezi mým novým tvarem a původním rozkladem.

Závěr mé práce vypovídá o tom, jak bylo mé testování úspěšné a čeho jsem dosáhl. Sděluji zde zhodnocení výsledků a celkový průběh. Dále v této části vyzdvihuji nové poznatky a skutečnosti, které jsem během celého vypracování tématu nabyl.

2 Kvantování

2.1 Úvodní poznámky

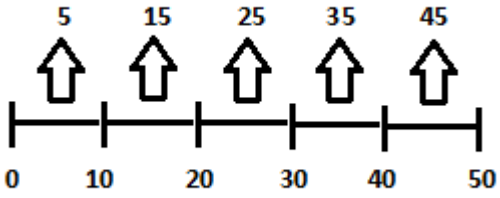
Kvantování je často základem ztrátových kompresních metod. V principu se jedná o diskretizaci oboru hodnot. Obor hodnot se rozdělí na intervaly, jimž je pak přidělena jediná zástupná hodnota. Jinak řečeno, vezme se „spojitá“ množina (rozsah) hodnot a reprezentuje se (nepřesně, se ztrátou informace) podmnožinou, která je tvořena hodnotami souvisejícími s původní množinou. [1]

Při kvantování dochází ke *kvantizační chybě*, která se v počítačové grafice projevuje například na plochách s malou změnou gradientu jako náhlý skok barev. Původně hladký přechod je nahrazen skokovou změnou a v obraze vznikají hrany. Neuniformní kvantování může kvantizační chybu částečně eliminovat. [1]

Dle typu kvantované veličiny se rozeznává:

- **kvantování skalárů** (*Scalar Quantization, SQ*) – vezme se jedna vstupní hodnota a výsledkem je jedna kvantovaná (kvantizační) výstupní hodnota

- **kvantování vektorů** (*Vector Quantization, VQ*) – vezme se skupina vstupních hodnot (vektor) a výsledkem je skupina kvantovaných (kvantizačních) výstupních hodnot (vektor)



Obr. 1: Princip kvantování

2.2 Kvantování vektorů

Vektorová kvantizace je proces, při kterém nekvantujeme jeden, ale několik obrazových bodů (blok 1x2, 1x4, 2×2, 4×4, 2×3,…) současně. Ty můžeme chápat jako vektor. Pro vstupní množinu všech vektorů pak hledáme jejich reprezentativní skupinu *k* vektorů tak, aby co nejlépe aproximovaly původní obrázek. Nalezené vektory se nazývají centroidy a tvoří kódovou knihu. Výhoda adaptivního hledání je v tom, že reprezentanti jsou optimálně nalezeni pro vstupní množinu dat. VQ dokáže využít a z velké části zachovat vzájemný vztah mezi daty

(pixely). Pro danou míru komprese dosahuje menší distorze, a naopak při dané distorzi dosahuje lepší míry komprese. [1]

Kvantování vektorů může být použito jako standardní kompresní metoda pracující s původními daty (obrázky) nebo může být použito jako kvantizační část některé obecné ztrátové komprese. [1]

2.2.1 Základní princip

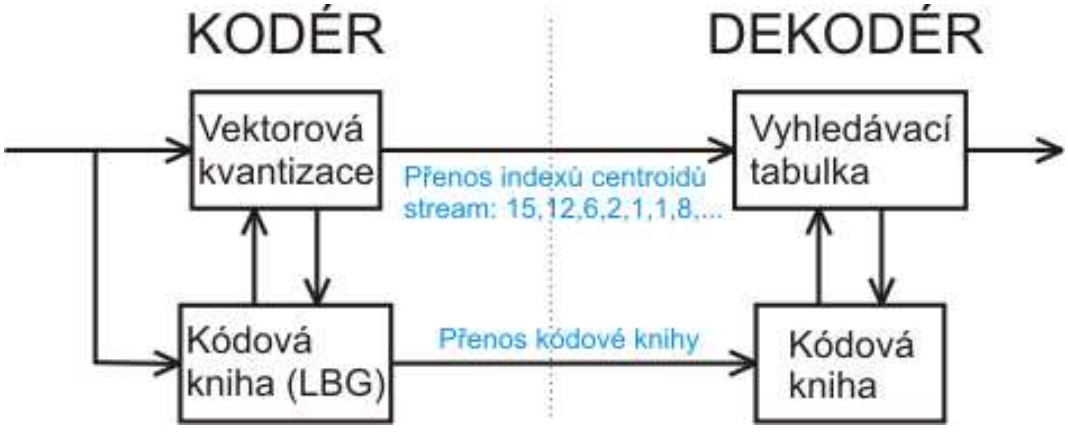
- Vytvoření slovníku (*codebook*). Ten se skládá z vektorů (kódových) o *n* prvcích (z bloků o *n* pixelech). [2]
- Kódování [2]

  1. Rozdělení vstupního obrazu na bloky (vektory) i *n* axelech
  2. Pro kazdý takto vzniklý vektor *v* se najde ve slovníku nejblizší kódový vektor v^*. Kóduje se vektor *v* indexem vektoru v^* ve slovníku.
  3. Bezeztrátová komprese získaných indexů.

- Dekódování (vyhledávací tabulka (LookUp Table, LUT)) [2]

  1. Bezztrátová dekomprese indexů.
  2. Nahrazení každého indexu příslušným kódovým vektorem v^* ze slovníku.

- Slovník je uložen nebo přenesen.
- Slovník může být vytvořen pro konkrétní obrázek nebo pro skupinu obrázků. Může tedy být „lokální“ nebo „globální“.



Obr. 2: Princip kvantování vektorů

Při kvantování vektorů se musí řešit několik otázek:

- velikost slovníku - N_c
- velikost bloku (vektorů) – n
- konstrukce slovníku
- struktura slovníku
- slovník pro jeden obrázek nebo skupinu obrázků

2.2.2 Velikost slovníku

- velký slovník umožňuje lépe zachytit jednotlivé rysy, vede k lepší kvalitě výsledného obrázku, ale je náročnější na uložení nebo přenos
- malý slovník má opačné vlastnosti
- typické velikosti slovníku jsou: 64, 128, 256, 512, 1024, 2048

2.2.3 Velikost bloků

- větší vektor lépe využívá vzájemný vztah mezi pixely
- zde záleží na tvaru bloku - 2x2, 4x4 (16 prvkový vektor), 1x4, 2x3, 2x4 atd.

2.2.4 Konstrukce slovníku

Kódovou knihu hledáme na základě některého kritéria, např. minimalizace střední kvadratické chyby. Její nalezení je hlavním problémem při vektorové kvantizaci, protože je to úloha nejednoznačná a navíc výpočetně velmi náročná. Nástrojem pro konstrukci slovníku je LBG algoritmus (*Linde-Bazo-Gray Algorithm*). [1]

Kroky LGB algoritmu: [3]

1. Začíná se počátečním slovníkem, který má N_c kódových vektorů
2. Vytvoří se N_c tříd z trénovacích vektorů – každý trénovací vektor v se přiřadí do třídy i, jestliže i-tý kódový vektor v^* je trénovacímu vektoru v nejbližší. (Trénovací množina je množina všech bloků získaných z obrázku určeného ke kompresi).
3. Opakovaně se přetváří jednotlivé třídy tak, že se pro každou třídu spočítá nový centroid (střed) v^*. Potom se pro každý trénovací vektor v najde nejbližší centroid v^* a trénovací vektor se přiřadí do třídy, kterou tento centroid v^* reprezentuje.

4. Tento proces se ukončí v okamžiku, když se celková oddychla (rozdíly mezi trénovacími vektory a jejich centroidy) již příliš nemění.

5. Jako slovník se vezmou nejnovější centroidy.

Protože kódová kniha obsahuje k centroidů, potřebujeme pro přenos jednoho $\log_2(k)$ bitů. Zároveň ovšem musíme myslet i na to, že musíme přenášet také kódovou knihu (jedná se o adaptivní algoritmus), která komprimovaná není a snižuje tak možný kompresní poměr. [3]

Vlastní kvantování je naopak úloha velmi snadná – každý vektor nahradíme nejbližším centroidem. Je zřejmé, že čím více bude centroidů, tím menší bude kvantizační chyba. Do souboru pak ukládáme pouze indexy centroidů, ke kterým byly jednotlivé vektory mapovány. Tím dochází ke značné úspoře místa. [1]

2.2.5 Inicializace slovníku

Pro vytvoření počátečního slovníku (inicializaci slovníku) se používá několik metod: [3]

- náhodný výběr
- rozdělování
- párování (shlukování) nejbližších sousedů

2.2.6 Struktura slovníku

Po vytvoření optimálního slovníku, tedy po vytvoření počátečního slovníku a jeho následné optimalizaci LBG algoritmem, dochází k vlastnímu kvantování vektorů. Jak již bylo dříve zmíněno, probíhá tak, že se pro každý vektor z trénovací množiny najde nejbližší kódový vektor ze slovníku a jeho index se potom zpracuje. [1]

Úloha nalezení nejbližšího kódového vektoru se nazývá NN problém (*Nearest Neighbor problem*). Náročnost nalezení nejbližšího vektoru závisí na struktuře slovníku. V obecném případě je slovník tabulkou (polem). Jinou možností je reprezentovat slovník stromem. [1]

Podle struktury se potom úloha nalezení nejbližšího kódového vektoru řeší jako:

- FSVQ (*Full Search Vector Quantization*) – pro pole
- TSVQ (*Tree Structured Vector Quantization*) – pro strom

FSVQ

Naše testovací aplikace řeší úlohu tímto způsobem, proto si ji stručně popíšeme. Hledání v poli je jednoduchou metodou Vezme se trénovací vektor, postupně se prochází slovník (pole) a hledá se nejbližší kódový vektor. Až se najde, jeho index se zpracuje.

Náročnost tohoto přístupu je: pro slovník o velikosti N_c se musí pro daný trénovací vektor provést N_c porovnání. Tedy například u slovníku o velikosti 256 je to 256 porovnání pro každý trénovací vektor. Je tudíž potřeba relativně mnoho porovnání.

2.2.7 Poznámky ke kompresi pomocí VQ

- Jestliže má slovník velikost N_c, je potřeba při dekvantování $\log_2 N_c$ bitů pro určení odpovídajícího kódového vektoru.

- V případě přenosu se počítá s tím, že slovník je dostupný na obou stranách, tudíž nejsou potřeba žádné bity pro přenos slovníku.

- Kompresní poměr vypočítáme jako počet bitů/pixel v komprimovaném obraze ku počtu bitů/pixel v nekomprimovaném obraze

2.3 Shrnutí

Tato kapitola dává dostačující obrázek o kvantování a hlavně o konkrétním typu - kvantování vektorů. Postup VQ je popsán solidním základem pro další praktickou implementaci.

3 Typy rozkladů

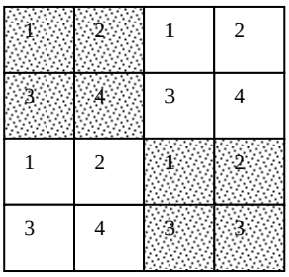
Na základě teorie o VQ jsem mohl navrhnout vhodné rozklady obrazových dat. Každý rozklad je něčím specifický a proto si jednotlivé typy musíme důkladně popsat a ukázat. Všechny návrhy jsou zmíněny ve stejném měřítku (1 čtverec = 1 pixel obrazu). V textu níže nám nové typy slouží k teoretickému popisu pro budoucí praktickou implementaci v ukázkové aplikaci. Dá se tedy předpokládat, že z důvodu různých rozlišení obrázků a nepravidelných tvarů v blocích, nám pravděpodobně nevznikne přesný rozklad rozlišení čtverce (jsou testovány pouze obrazové data se stejnou šířkou a výškou). Proto bude testovací program nachystán tak, že zbývající části obrázku ořízne a nebude se jimi zabývat.

VQ provádí kompresi po blocích, proto v mé aplikaci testuji obrázky na bloky různých rozměrů. Vektor pro kvantizaci nemusí mít vždy velikost bloku 2x2=4, 4x4=16, 8x8=64, ale vektor pro naše jiné rozklady může mít velikosti 3, 4, 5, 6, 12, 20. V následujícím popisu bude vyznačeno, jak jsem jednotlivé typy návrhu poskládal do bloků a jaký rozměr tato část má. Taky si postupně očíslujeme pixely, jak jdou jako vektor v bloku přesně za sebou.

3.1 Čtvercový rozklad

Standardní rozklad obrázků při kompresi pomocí VQ. Poslouží mi jako základ pro další nové typy. Každý čtvereček „odpovídá“ jednomu pixelu v obrazu. Pro budoucí práci důležitý rozklad, všechny ostatní navržené typy otestuji, a pak budu s tímto čtvercovým porovnávat a hodnotit rozdíly a odlišnosti, případně stejné nebo lepší rysy.

Obrázek rozkládáme v blocích postupně pěkně v řadě za sebou pixel po pixelu. Jakmile dojdeme na konec bloku v sloupci, vrátíme se na začátek o řádek níž. Takto procházíme po částech celý tvar, dokud nenarazíme na konec bloku v řádku. Tím se vytvoří celý výsledný vektor a předává se dále. Rozklad je složen s několika různých bloků, nejčastěji se budeme zabývat bloky 2 x 2 (vektor délky 4) a 4 x 4 (vektor délky 16). Na následujících obrázcích lze vidět výsledný rozklad dle této metody, čísla udávají pořadí, jak jsou pixely zapsány jako posloupnost vektoru.



Obr. 3a: Čtvercový rozklad obrázku, délka vektoru 4 (2x2)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Obr. 3b: Čtvercový rozklad obrázku, délka vektoru 16 (4x4)

3.2 Obdélníkový rozklad

Prvním z mých navržených nových typů rozkladu obrázků bude jednoduché rozložení obrazových dat na obdélníky. Tento způsob jsem vybral na úvod záměrně, jedná se asi o nejjednodušší variantu mých návrhů. Nejde o nic jiného, než o mírně upravený rozklad čtvercový. Velikost bloku závisí na délce obdélníku. Pro testování byl navrhnut vybrán obdélník o velikosti 4 (délka vektoru = 4) a 16.

Princip tohoto typu spočívá pouze v jednom kroku. V bloku neprocházíme obrázek postupně za sebou pixel po pixelu. Potřebujeme dostat tvar obdélníku, proto se posunujeme na počáteční pixely tvaru a s ním hned zapíšeme do vektoru všechny pixely v obdélníku o zvolené délce. Tímto způsobem projdeme celý námi zvolený blok a předáme dosažený vektor dále. Následující obrázek přesně ukazuje tento rozklad, jednotlivé tvary jsou pro přehlednost odděleny vzorkováním.

Poznámka - U obrázku s tvarem o velikosti vektoru 16 byl z důvodu nedostatku místa rozklad proveden na dva řádky (ve skutečnosti je to řádek jeden).

1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4

Obr. 4a: Obdélníkový rozklad obrázku, délka vektoru 4

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16

Obr. 4b: Obdélníkový rozklad obrázku, délka vektoru 16

3.3 Trojúhelníkový rozklad

Dalším z řady nových tvarů pro rozklad jsem vybral metodu rozložení obrázků na klasické trojúhelníky. Tento způsob je už o trošku složitější a setkáme se zde s otočením tvaru. Musíme se snažit, aby do sebe bloky vhodně „zapadly“. Návrh se bude skládat ze dvou typů tohoto tvaru. První rozklad je složen ze tří pixelů, velikost bloku a vektoru je tedy 3. Ve druhém návrhu si tvar v stejném měřítku zvětšíme a pokryjeme tím větší plochu obrazu. Vznikne nám blok o délce vektoru 12.

Základním problémem toho rozkladu je nepravidelnost tvaru. Jelikož používáme VQ (komprese v blocích) musíme tvary vhodně poskládat a zbylé místo v obrázku, které už blok nepokryje, nebude do procesu VQ zahrnuto. Z následujícího obrázku je patrné, jak tento rozklad vypadá. Při otáčení tvaru je důležité zachovat stejnou posloupnost pixelů v bloku (první pixel v klasickém tvaru odpovídá prvnímu pixelu v obráceném tvaru), protože z tvarů opět zpětně skládáme obrázky.

1	2	3	1	2	3
3	2	1	3	2	1

Obr. 5a: Trojúhelníkový rozklad obrázku, délka vektoru 3

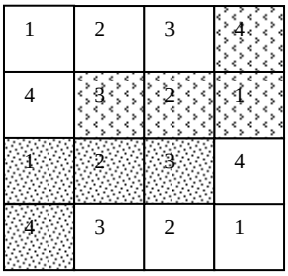
1	2	3	4	12	11
5	6	7	8	10	9
9	10	8	7	6	5
11	12	4	3	2	1

Obr. 5b: Trojúhelníkový rozklad obrázku, délka vektoru 12

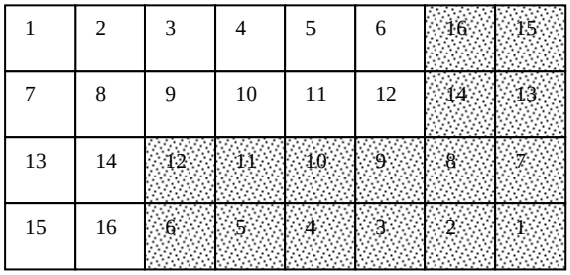
3.4 „L“ rozklad

Pří sestavování dalšího typu rozkladu jsem lehce upravil trojúhelníkový rozklad a navrhl jsem ještě jednu podobnou metodu rozložení obrázku. Jedná se o tvar připomínající písmeno L. Výhodou tohoto návrhu je, že tvary jsou pravidelné a nebudeme muset nic při VQ vynechávat.. Základní tvar se skládá ze čtyř pixelů, tudíž velikost vektoru je 4. Rovněž i zde si ukážeme rozklad, který zabírá větší blok v obrázku a bude se skládat dokonce z 16 pixelů. Velikost vektoru bude tedy 16, shodou okolností stejná velikost jako u čtvercového rozkladu a bloku 4x4.

Tento způsob rozložení má vždy dvě části. V prvním kroku se symbol položí pěkně za sebou na délku, v druhém kroku se tvary položí stejným způsobem, ale zrcadlově obráceně. Tímto nám obě části do sebe zapadnou. Takto spojené tvary do sebe budeme využívat i v druhé verzi návrhu. Opět je důležité u převráceného tvaru zachovat původní posloupnost zapsaných pixelů do vektoru. Na obrázku vidíme výsledné tvar a rozložení.



Obr. 6a: „L“ rozklad, vektor délky 4



Obr. 6b: „L“ rozklad, vektor délky 16

3.5 Schodový rozklad

Předposlední návrh se zabývá symbolem klasických schodů, tak jak je známe. Jedná se na implementaci už o složitější tvar, ale opět jeho velká výhoda spočívá v tom, že nám tvary do sebe i po otočení krásně zapadnou. Tyto bloky nám nepravidelně rozloží čtvercový obraz, tudíž znovu musíme některé zbylé části vynechat při VQ.

Základní symbol obsahuje celkem šest pixelů, které jsou na sobě naskládány do tří řad tvořící schody (každá následující řada má o jeden pixel méně). Velikost bloku tohoto rozkladu je tedy 6 (3 + 2 + 1). V druhém větším tvaru je symbol rozšířen způsobem, který nám prodlužuje délku každého schodu. Proto tento tvar má délku vektoru 15 a pět řad.

Nyní je sestavování rozkladu složeno celkem na dvě části. V prvním kroku nachystám tvary schodů vzhůru nohama tak, že nejdelší řada bude úplně nahoře a do vektoru zapíši všechny souřadnice. V druhém kroku už jde jen o to, aby tvar zapadl přesně do nachystaných stupínků z předchozí části. Situaci si představme a vznikl nám v základním tvaru celkový blok o

velikost 4x3. Tímto způsobem budeme v obrázku pokračovat při jeho rozložení. Pro lepší znázornění opět výsledné rozklady obrázků a ekvivalentní očíslování pixelů v obou částech.

1	2	3	6	1	2	3	6
4	5	5	4	4	5	5	4
6	3	2	1	6	3	2	1

Obr. 7a: Schodový rozklad, vektor délky 6

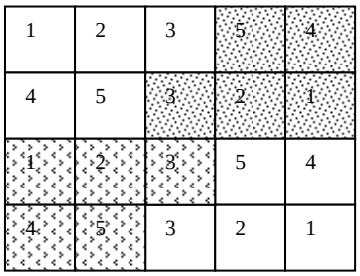
1	2	3	4	5	15
6	7	8	9	14	13
10	11	12	12	11	10
13	14	9	8	7	6
15	5	4	3	2	1

Obr. 7b: Schodový rozklad, vektor délky 15

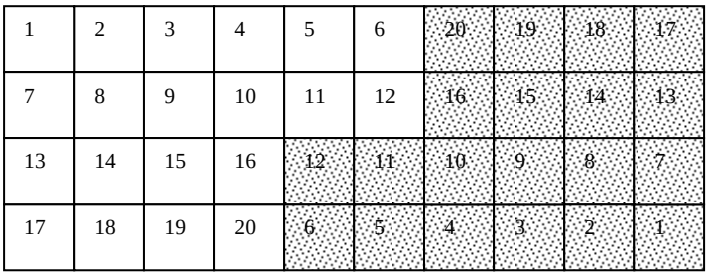
3.6 Schodový rozklad II

Dostáváme se k poslednímu navrženému rozkladu. Jedná se o podobný symbol jako z předchozí ukázky. Hlavní rozdíl je jen v jedné věci. Chybí na tomto tvaru horní vrchol schodu (1 pixel). Z toho tedy můžeme soudit, že základní tvar se skládá z 5 pixelů a obsahuje dvě řady na sobě. Délka vektoru, který nám blok zaplní je tedy 5. I zde si ukážeme modifikaci tohoto základního tvaru. Jedná se o zvětšení všech stran o dvojnásobek a dostaneme blok o délce a velikosti vektoru 20.

Na rozložení bloku je tento návrh podobný jako schodový rozklad. Rozdělil bych ho na dvě části. V prvním kroku se tvar v bloku otočí na šířku vzhůru nohama. Tím nám vznikne část, která nám nabízí prostor pro zapadnutí druhého tvaru. V následujícím kroku do vzniklého místa vložíme další tvar, otočený sice na šířku, ale už v poloze připomínající schod. Takto tedy zaplníme celý požadovaný blok a rozklad je hotov. Tyto nepravidelné tvary a z nich vyplývající bloky, nám při implementaci nezaplní dokonale čtvercový obrázek. Proto bude opět obraz ořízнut a drobné části nevyužijeme pro VQ. Na následujících obrázcích lze vidět oba výsledné rozklady a posloupnost vektoru v bloku.



Obr. 8a: Schodový rozklad II, vektor délky 5



Obr. 8b: Schodový rozklad II, vektor délky 20

3.7 Shrnutí

V celé kapitole jsme si na základě znalostí VQ důkladně popsali jak původní čtvercový rozklad, tak i nové typy rozkladu. U každého rozložení jsem dodal charakteristiku daného typu, slovní popis a možnosti návrhu. Nezbytnou součást pro představu každého návrhu je obrázek s vyznačenými symboly a očíslováním jednotlivých pixelů, jak jdou v řadě za sebou. Nyní máme vše připraveno k tvorbě ukázkové aplikace, která nám otestuje tyto teoretické návrhy.

4 Aplikace Compress

Aplikace Compress slouží pro praktickou ukázku komprese obrazových dat pomocí kvantování vektorů a využívající různé typy rozkladů obrázků. Při její tvorbě byl použit nástroj Microsoft Visual Studio 2008 a programovací jazyk C++. Z důvodu rychlejšího zpracování kódu jsou v tomto programovacím jazyce napsány i hlavní algoritmy a metody rozkladů.

Obr. 9: Aplikace Compress

4.1 Prostředí aplikace

Jednotlivé hlavní části aplikace jsou:

- **Form1 –** hlavní formulář aplikace „main“, ze kterého se spouští a zobrazují ostatní části programu, obsahuje menu, 3 PictureBoxy (2 slouží pro zobrazení obrazu, jeden je zaveden v porovnání rozdílů) a spouštěcí tlačítko
- **MainMenu –** nabídka celé aplikace, obsahuje načtení obrázku, nový obrázek, ukončení aplikace, nastavení komprese (Form2) a informace o programu

- **Form2 –** slouží k nastavení všech parametrů pro kompresi (velikost bloku a kódové knihy, nastavení typu rozkladu, nastavení délky vektoru)
- **PBOriginal –** místo pro zobrazení načteného původního obrázku
- **PBAfterCompres –** po kompresi se do tohoto boxu zobrazí výsledný obraz
- **PBDifferences –** pracovní box, slouží k rozlišení rozdílů od původního a výsledného obrázku
- **Třída data –** zde jsou implementovány metody načtení obrázků, nastavení parametrů bloků, rozložení bloků dle tvaru, spuštění algoritmu komprese, zpětné složení bloků, vykreslené rozdílového obrazu
- **Třída Alg –** zde implementovány kompresní algoritmy

V naší aplikaci jsou kromě funkcí, které s ním souvisí i další funkce, které se starají o vykonání kompresního algoritmu nebo o rozložení obrazu zvoleným novým rozkladem. Mají tudíž podobné názvy. Při práci s obrázky jsou funkce ukončeny písmeny RGB.

Celý proces komprese se provádí 3x pro každou barvu RGB zvlášť a skládá se z těchto kroků:

1. Nastavení bloků
2. Rozložení obrázku na bloky dle tvaru
3. Inicializace (naplnění) slovníku
4. Tvorba slovníku
5. Kvantování
6. Zpětné složení obrázku dle tvaru
7. Vykreslení výsledného obrázku
8. Vykreslení rozdílového obrazu
9. Zobrazení výsledků

Zpracování obrázků probíhá obvykle v těchto krocích:

1. Načte se obrázek do přiděleného boxu (MainMenu – File -- Open)
2. Nastaví se parametry pro kompresi (Form2)
3. Provede se komprese (kvantování)

4. Zobrazí se výsledný obrázek
5. Zobrazí se obrázek rozdílů
6. Zobrazí se výsledky, počet změn pixelů, procenta změn a Euklidovská vzdálenost rozdílů dle vzorce:

$$EV = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$$

(kde *x* a *y* jsou hodnoty pro RGB složky původního a komprimovaného obrázku)

4.2 Shrnutí

V této kapitole jsme si stručně přiblížili vzorovou aplikaci, na které budeme testovat navržené typy rozkladu a zkoumat dosažené výsledky. Popsali jsme si prostředí aplikace, všechny důležité vizuální prvky hlavní části programu. K nim patří i rozbor potřebných a nezbytných atributů používaných pro kompresi dat. Nakonec jsme si přiblížili, jak program postupuje v jednotlivých krocích a jak zpracovává postupně obrázek.

Kvantování vektorů je relativně náročný proces. Rychlost aplikace záleží na velikosti a složení obrázku, nastavených parametrech komprese a výkonu PC.

5 Testování rozkladů

Celá tato kapitola se zabývá testováním navržených rozkladů pro kompresi dat pomocí aplikace Compress. I když bylo testování prováděno na daleko větším a pestřejším počtu obrázků, pro přiblížení dosažených výsledků u navržených rozkladů byli vybrání tito zástupci (komprese byla prováděna na barevné podobě). Všechny ostatní tabulky s výsledky testování i použité obrázky jsou dodány v příloze k této práci.

Obr. 10: Zkušební obrázky – lena, house

Při samotném testování se sledovalo množství různých parametrů, které nám mohly poskytnout ucelený přehled o samotné kvalitě výsledného testu na navržený rozklad pro kompresi. Jednotlivé výsledky jsou zaznamenány v tabulkách. Před každým testem je nutné nastavit typ rozkladu, délku vektoru v rozkladu, práh a velikost slovníku.

V tabulkách jsou pozorovány tyto parametry:

1. U vybraného obrázku sledujeme celkový počet pixelů. Po provedení testu je uveden rozdíl v procentech, to znamená, v kolika pixelech se původní a výsledný obrázek liší. Vysoké hodnoty ještě neznamenají špatný výsledný obraz, protože lidské oko téměř nezaznamená drobné změny.

2. Po kompresi je dále vypočítána Euklidovská vzdálenost (EV) výsledného obrazu od původního obrázku. Na základě hodnot RGB se v tabulce objevuje průměrná hodnota těchto tří složek. Tato hodnota nám tedy říká to, že čím větší rozdíly v obrazu, tím větší euklidovská vzdálenost a naopak. Proto tohle číslo bude daleko přesnější a důležitější, než objektivní rozdíl změn v procentech, a bude na hodnotu EV kladen důraz při vyhodnocování výsledků.

5.1 Testy obdélníkového rozkladu

Obdélníkový rozklad otestujeme následovně. Na naše zástupce obrázků vyzkoušíme kompresi pro všechny velikosti slovníku a dostupné délky vektoru 4 a 16. V tabulkách jsou vidět dosažené výsledky testování.

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	4	0.08	99.92	2145.56
		16	0.03	99.97	3998.58
128	0.01	4	0.17	99.83	1763.42
		16	0.09	99.91	3498.67
256	0.01	4	0.32	99.68	1491.17
		16	0.13	99.87	3030.21
512	0.01	4	0.61	99.39	1274.21
		16	0.46	99.54	2590.98
1024	0.01	4	1.13	98.87	1065.98
		16	2.07	97.93	2086.81
2048	0.01	4	2.18	97.82	873.71
		16	11.98	88.02	1487.23

Tabulka 1: Obdélníkový rozklad na obrázek lena

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	4	0.11	99.89	2218.58
		16	0.03	99.97	3999.82
128	0.01	4	0.26	99.74	1824.91
		16	0.09	99.91	3448.72
256	0.01	4	0.53	99.47	1512.91
		16	0.19	99.81	2980.26
512	0.01	4	0.94	99.06	1249.57
		16	0.78	99.22	2521.55
1024	0.01	4	2.27	97.73	1071.69
		16	2.47	97.53	2067.21
2048	0.01	4	3.72	96.28	874.56
		16	12.73	87.27	1540.41

Tabulka 2: Obdélníkový rozklad na obrázek house

Kvalita obrazu závisí na zvolených parametrech, hlavně na velikosti slovníku. Tudíž nejlepší nastavení se jeví při slovníku 2048. Rovněž délka vektoru hraje roli v kvalitě obrázku, při vektoru délky 4 je obraz vykreslen daleko přesněji.

5.2 Testy trojúhelníkového rozkladu

Trojúhelníkový rozklad otestujeme podobně jako předchozí návrh. Na naše zástupce obrázků vyzkoušíme kompresi pro všechny velikosti slovníku a dostupné délky vektoru 3 a 12. V tabulkách jsou vidět dosažené výsledky testování.

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	3	0.55	99.45	1606.48
		12	1.65	98.35	2940.37
128	0.01	3	0.64	99.36	1301.65
		12	1.71	98.29	2594.37
256	0.01	3	0.87	99.13	1067.61
		12	1.77	98.23	2319.49
512	0.01	3	1.43	98.57	878.51
		12	1.88	98.22	2031.91
1024	0.01	3	2.73	97.27	727.48
		12	2.31	97.69	1776.51
2048	0.01	3	5.43	94.57	605.15
		12	6.96	93.04	1445.92

Tabulka 3: Trojúhelníkový rozklad na obrázek lena

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	3	0.51	99.49	2011.99
		12	1.59	98.41	3733.29
128	0.01	3	0.71	99.29	1562.53
		12	1.68	98.32	3256.22
256	0.01	3	1.16	98.84	1267.72
		12	1.84	98.16	2895.33
512	0.01	3	1.74	98.26	1058.37
		12	2.04	97.96	2540.99
1024	0.01	3	3.45	96.55	863.31
		12	2.85	97.15	2170.11
2048	0.01	3	6.06	93.94	719.87
		12	7.41	92.59	1681.02

Tabulka 4: Trojúhelníkový rozklad na obrázek house

Kvalita obrazu závisí na zvolených parametrech, hlavně na velikosti slovníku. Tudíž nejlepší nastavení se jeví při slovníku 2048, kde vidíme u vektoru délky 3 skoro dokonalé vykreslení. Tudíž vektor hraje roli v kvalitě obrázku, při vektoru délky 3 je obraz vykreslen daleko přesněji.

5.3 Testy „L“ rozkladu

„L“ rozklad můžeme otestovat přesně stejně jako rozklad obdélníkový. Tvary mají stejnou hodnotu vektoru, tudíž bude pro nás zajímavé dokonalé porovnání těchto dvou návrhů a jejich dosažených kvalit. Testujeme vektory délky 4 a 16, výsledky jsou v následujících tabulkách.

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	4	0.10	99.90	2025.79
		16	0.04	99.96	3323.41
128	0.01	4	0.20	99.80	1718.69
		16	0.12	99.88	2934.11
256	0.01	4	0.32	99.68	1447.73
		16	0.23	99.77	2596.53
512	0.01	4	0.58	99.42	1228.79
		16	0.45	99.55	2321.29
1024	0.01	4	1.01	98.99	1053.94
		16	2.14	97.86	1899.93
2048	0.01	4	2.29	97.71	868.95
		16	11.84	88.16	1409.81

Tabulka 5: „L“ rozklad na obrázek lena

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	4	0.08	99.92	2468.89
		16	0.06	99.94	4159.96
128	0.01	4	0.22	99.78	2012.81
		16	0.09	99.91	3647.08
256	0.01	4	0.44	99.56	1665.15
		16	0.23	99.77	3179.43
512	0.01	4	1.01	98.99	1412.52
		16	0.72	99.28	2706.33
1024	0.01	4	1.75	98.25	1176.91
		16	2.37	97.63	2264.79
2048	0.01	4	3.21	96.79	985.56
		16	11.82	88.18	1714.58

Tabulka 6: „L“ rozklad na obrázek house

Obrázek opět závisí na zvolených parametrech. Nejlepší nastavení se jeví při slovníku 2048. Rovněž délka vektoru hraje roli v kvalitě obrázku, při vektoru délky 4 je obraz vykreslen daleko přesněji. Porovnání tohoto rozkladu a obdélníkového je uvedeno v závěru kapitoly.

5.4 Testy schodového rozkladu

Schodový rozklad testujeme jako všechny předchozí tejným způsobem. Na obrázky provedeme kompresi a vektor bude nastaven na velikosti 6 a 15. Dosažené výsledky jsou zaznamenány v tabulkách.

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	6	0.51	99.49	2345.36
		15	2.01	97.99	3175.28
128	0.01	6	0.59	99.41	2027.55
		15	2.06	97.94	2871.56
256	0.01	6	0.72	99.28	1778.29
		15	2.14	97.86	2586.71
512	0.01	6	0.87	99.13	1562.41
		15	2.34	97.66	2352.75
1024	0.01	6	1.09	98.91	1335.87
		15	3.69	96.21	1994.35
2048	0.01	6	1.88	98.22	1150.01
		15	11.51	88.49	1656.74

Tabulka 7: Schodový rozklad na obrázek lena

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	6	0.45	99.55	2982.66
		15	2.01	97.99	4098.18
128	0.01	6	0.59	99.41	2580.89
		15	2.06	97.94	3658.36
256	0.01	6	0.71	99.29	2206.01
		15	2.15	97.85	3251.83
512	0.01	6	0.89	99.11	1934.46
		15	2.41	97.59	2864.31
1024	0.01	6	1.39	98.61	1627.11
		15	3.84	96.16	2396.25
2048	0.01	6	2.75	97.25	1368.83
		15	12.02	87.98	1939.68

Tabulka 8: Schodový rozklad na obrázek house

Složitost tohoto tvaru se podepsala na výsledku testování. Hodnoty EV jsou celkem vysoké a obrázek není zdaleka tak přesně vykreslen jak u předchozích návrhů. I přes to nejlepší nastavení najdeme opět při délce vektoru 6 a slovníku 2048.

5.5 Testy schodového rozkladu II

Dostáváme se k poslednímu rozkladu a testování. Bez větších změn je způsob stejný jako v předchozích případech, jen velikost vektoru použitá při kompresi je 5 a 20. V tabulkách jsou opět zaznamenány dosažené výsledky.

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	5	0.53	99.47	2195.89
		20	2.43	97.57	3400.85
128	0.01	5	0.58	99.42	1899.72
		20	2.48	97.52	3021.92
256	0.01	5	0.67	99.33	1649.96
		20	2.61	97.39	2724.29
512	0.01	5	0.87	99.13	1417.43
		20	3.19	96.81	2360.31
1024	0.01	5	1.26	98.74	1231.45
		20	6.26	93.74	1958.19
2048	0.01	5	1.92	98.08	1024.19
		20	20.77	79.23	1366.39

Tabulka 9: Schodový rozklad II na obrázek lena

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	5	0.45	99.55	2611.21
		20	2.38	97.62	4117.56
128	0.01	5	0.61	99.39	2200.12
		20	2.48	97.52	3682.18
256	0.01	5	0.77	99.23	1872.73
		20	2.55	97.45	3174.46
512	0.01	5	1.27	98.73	1595.26
		20	3.22	96.78	2797.96
1024	0.01	5	1.75	98.25	1374.15
		20	6.57	93.43	2258.98
2048	0.01	5	3.17	96.83	1148.06
		20	20.88	79.22	1594.55

Tabulka 10: Schodový rozklad II na obrázek house

Poslední výsledky testování ovlivnilo složitější poskládání tvaru. Zvlášť při vektoru 20 a malé velikosti slovníku je obrázek rozmazaný, čemuž napovídají hodnoty EV. Nejlepší výsledek byl dosáhnut při vektoru 5 a slovníku 2048.

5.6 Shrnutí

Z provedených testů na rozkladech vyplývá:

- kvalita výsledného obrázku závisí na nastavených parametrech (hlavně velikost slovníku a délka vektoru) a typu obrázku
- čím jednodušší tvar, tím lepší výsledný obraz
- nejlepšího nastavení u každého rozkladu dosáhneme při velikosti slovníku 2048
- rozdílové hodnoty EV schodového rozkladu byly jednoznačně nejvyšší (horší kvalita obrázku) i při vyšších velikostech slovníku
- naopak nejlépe dopadl rozklad trojúhelníkový, hlavně z důvodu malé velikosti vektoru (délka 3)
- v přímém porovnání obdélníkového a „L“ rozkladu (vektory jsou si v obou případech rovny) jsou u obrázku lena podobné hodnoty EV, při obrázku house ale lépe kompresi zvládl obdélníkový rozklad
- celkově u všech rozkladů nejsou značné výkyvy ve výsledných obrázcích, nedá se tedy říci, že by byl některý horší nebo lepší, vše záleží na parametrech a typu obrázku

6 Porovnání rozkladů

V poslední kapitole naší práce se podíváme na porovnání otestovaných rozkladů se standardním čtvercovým rozkladem. Možnosti porovnání jsou omezeny délkou vektoru u čtvercového rozkladu. Pro něj jsou použity délky 4 (2 x 2) a 16 (4 x 4). Před samotným srovnáváním je tedy nutné provést testy i na tomto standardním typu, abychom měli z čeho vycházet.

6.1 Testy čtvercového rozkladu

Testování provedeme pro vektory 4 a 16 stejným způsobem jako u jiných rozkladů.

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	4	0.16	99.84	1895.37
		16	0.11	99.89	3095.74
128	0.01	4	0.29	97.71	1576.45
		16	0.15	99.85	2770.06
256	0.01	4	0.36	99.64	1324.59
		16	0.25	99.75	2490.83
512	0.01	4	0.62	99.38	1130.83
		16	0.57	99.43	2154.43
1024	0.01	4	1.31	98.69	970.53
		16	2.19	97.81	1815.35
2048	0.01	4	2.63	97.37	801.99
		16	11.20	88.80	1369.18

Tabulka 11: Čtvercový rozklad na obrázek lena

Velikost Slovníku	Práh	Délka vektoru	Nezměněné pixely (%)	Změněné pixely (%)	Hodnota EV
64	0.01	4	0.12	99.88	2234.07
		16	0.05	99.95	3840.81
128	0.01	4	0.35	99.65	1868.09
		16	0.17	99.83	3425.73
256	0.01	4	0.41	99.59	1558.29
		16	0.29	99.71	3029.29
512	0.01	4	0.85	99.15	1302.67
		16	0.71	99.29	2644.36
1024	0.01	4	2.11	97.89	1101.89
		16	2.50	97.50	2197.73
2048	0.01	4	3.48	96.52	921.39
		16	12.37	87.63	1642.33

Tabulka 12: Čtvercový rozklad na obrázek house

6.2 Srovnání rozkladů

Nyní máme vše připraveno a můžeme porovnat hodnoty jednotlivých rozkladů se standardním čtvercovým obrázkem. Využijeme testovací výsledky z předchozích tabulek, kde se pěkně, hlavně v hodnotě EV dozvídáme, jak se daný rozklad projevil na obrázek, co se týče změn v obrazu po kompresi.

6.2.1 Čtvercový a obdélníkový

V tomto porovnání se důkladně podíváme na výsledky komprese. Máme k dispozici u obou rozkladů velikost vektoru 4 a 16, což nám dává přesný obraz o kvalitě. V hodnotách vidíme názorný příklad toho, jak záleží na typu obrázku. Zatím co u obrázku lena obdélníkový rozklad v kvalitě značně ztrácí (u obou vektorů jsou EV o dost vyšší), u obrázku house jsou hodnoty v obou vektorech velmi podobné. Dokonce je obrázek house vykreslen přesněji v nejlepším nastavení u obdélníkového rozkladu a délce vektoru 4.

Dostáváme tedy velmi zajímavý výsledek. Jak lze vidět, kvalita obdélníkového rozkladu může být v některých typech obrázků ve výsledku dokonce i lepší. V tomto směru je tedy obdélníkový rozklad dobrým konkurentem rozkladu čtvercového a můžeme být s výsledky spokojeni.

6.2.2 Čtvercový a trojúhelníkový

Porovnání těchto typů bude samozřejmě obtížnější. Rozklady využívají rozdílné délky vektoru, tak nám nezbývá nic jiného než předpokládat kvalitu na základě našich výsledků. Je třeba si uvědomit, že obrázky vypadají „na oko“ podobně, i když je většina pixelů změněna. Proto klademe důraz na hodnotu EV. Hodnoty u trojúhelníkového rozkladu vypadají tedy značně nižší (malý vektor délky 3). To nám ale dává nejasnou představu. Pokud se ale podíváme na výsledky u vektoru 12, jsou podobné a nakonec horší než vektor délky 16 u čtverce (obecně platí, čím menší vektor, tím lepší vykreslení).

Z tohoto poznatku by mělo být patrné, že nový rozklad nedosahuje takových kvalit při větším vektoru. Pro malý vektor je určitě výhodné rozklad použit, obrázek je po kompresi velmi kvalitní, ale pro program časově náročný proces při velkém slovníku.

6.2.3 Čtvercový a „L“

Další porovnání rozkladů nám poskytne opět přesný pohled na kvalitu, „L“ rozklad se skládá z vektorů délky 4 a 16, což je přesně to samé jako u rozkladu čtvercového. Zde se už ale projevuje složitější poskládání tvaru do sebe a na výsledku je to znát. U obou testovaných obrázků je čtvercový rozklad sice lepší, ale při velkém slovníku jsou rozdíly nepatrné. Problém nastává při malém slovníku u vektoru 16, kdy už je obrázek celkem značně deformovaný a zvýšené hodnoty EV to jen potvrzují.

Ve výsledku se „L“ rozklad při vektoru 4 dokáže přiblížit čtvercovému rozkladu. V jiných případech se projeví složitější složení tvaru a kvalita obrázku po kompresi není dostatečná.

6.2.4 Čtvercový a schodový

Porovnání těchto rozkladů bude o něco lehčí, tvar nabízí velikosti vektorů 6 a 15. Což je při větším vektoru přibližující pro čtvercový typ. Při testování se projevilo složitější poskládání tvaru schodu při velkém vektoru, kdy hodnoty EV jsou podobné a postupně větší při vektoru délky 15 než u vektoru čtverce délky 16. Tudíž v tomhle směru schodový rozklad značně zaostává. U menšího vektoru jsou hodnoty EV docela vysoké a nelze předpokládat kvalitní výsledek.

U schodového rozkladu je nejvíce poznat složitost tvaru. Výsledky nedosahují kvalitních hodnot a u menšího slovníku je obraz značně zdeformovaný. Proto v konečném hodnocení jasně vítězí standardní rozklad.

6.2.5 Čtvercový a schodový II

Poslední srovnání nám nabízí mírně upravený schodový rozklad. Zde bude pozorování opět relativní. Máme k dispozici vektory 5 a 20. Z tabulek lze vidět, že u většího vektoru jsou hodnoty podobné jak u čtvercového rozkladu. To nám značí nekvalitní rozklad (vektor 20 ku 16). Testy opět ovlivněny složitějším tvarem, při malém slovníku obraz nepřesně až rozmazaně vykreslen. Při malém vektoru jsou hodnoty EV vcelku příznivé a rozdíly k čtvercovému rozkladu by mohly odpovídat jednomu pixelu navíc v délce vektoru.

Ve výsledném zhodnocení je rozklad ovlivněn mírnou složitostí tvaru hlavně v malém slovníku. Při délce vektoru 5 je obrázek vykreslen podobně jako u čtvercového typu a velikosti vektoru 4. Tudíž tuto část lze hodnotit kladně.

6.3 Shrnutí

Ukázali jsme si porovnání všech rozkladů ku čtvercovému rozkladu. Obdélníkový rozklad se jeví v některých případech jako silný konkurent standardnímu rozkladu. Ostatní rozklady sice neukázaly lepší výsledky jako čtvercový rozklad. Nemůžeme ale říci, že je nějaký přímo horší a zbytečný. Čím složitější tvar, tím jsou výsledky horší a v porovnání s čtvercovým rozkladem už nemůžou nabývat lepších hodnot vykreslení. U rozkladů, jejichž vektory přímo neodpovídaly čtvercovým vektorům, jsme byli schopni i tak na základě dosažených testů srovnat kvalitu obrázků.

7 Závěr

V této bakalářské práci jsme se zabývali testováním obrazových dat při různém typu rozkladu.

V teoretické části práce bylo popsáno samotné kvantování. Uvedli jsme si základní popis a vlastnosti této techniky. Hlavní důraz byl kladen na princip kvantování vektorů, vlastnosti slovníku a postup provádění algoritmu. Pomocí této zajímavé techniky jsme pak provedli kompresi obrazových dat. Výklad byl zachycen pokud možno srozumitelně a také, aby získané znalosti bylo možno prakticky použít. Další informace lze získat v příslušné literatuře.

Následoval teoretický návrh rozložení obrazových dat. Vymyslel jsem vhodné nové typy rozkladu, které různými způsoby rozloží obrázek na bloky a tvary různých velikostí. K těmto všem těmto typům rozkladu byl popsán popis, vlastnosti a velikosti vektorů v bloku. Pro lepší představivost nechybí u každého nového typu obrázek rozkladu.

Pro praktické účely a testování byla vytvořena aplikace Compress, která slouží k demonstraci uvedených rozkladů obrázků. Zde jsem naimplementoval své rozklady a všechny metody s tím související. Aplikace je uživatelsky příjemná a umožňuje nejrůznější nastavení. Další zdokonalení není vyloučeno.

Pomocí aplikace byly následně provedeny testy. Na vybranou sadu obrázků se použily jednotlivé rozklady. Při různém typu nastavení jsme hodnoty zaznamenávali do tabulek. Po testování byly zhodnoceny výsledky ke každému rozkladu.

Z těchto poznatků jsme mohli uvedené rozklady porovnat se standardním čtvercovým rozkladem. Z praktických testů vyplývá, že i při jiném typu rozkladu lze dosáhnout velmi dobrých výsledků. Hlavně obdélníkový tvar má ke standardu nejblíže a v některých případech může být i lepším řešením.

Lze říci, že se podařilo splnit požadavky uvedené v zadání. Mé navržené rozklady dosáhly velmi zajímavých výsledků, a proto hodnotím práci a testování kladně. Jedinou nevýhodou je náročná kompresní část při kvantování vektorů. Z tohoto důvodu může program při velkém obrázku a velkém slovníku pracovat i několik minut.

Literatura

1. TŮMA, M. *Komprese obrazu – zpracování vektorů.* Olomouc, 1999. 53 s. Diplomová práce na Přírodovědecké fakultě Univerzity Palackého na katedře matematické informatiky. Vedoucí diplomové práce RNDr. Václav Snášel, CSs.

2. GRAY, Robert M. *Vector Quantization* [online]
   URL: < http://ee.stanford.edu/~gray/positano.pdf > [citováno 7. dubna 2010]

3. YOUSSEF, Abdou *Data Compression* [online]
   URL: < http://www.seas.gwu.edu/~ayoussef/cs225/> [citováno 7. dubna 2010]

Přílohy na CD

1. Zdrojové soubory aplikace Compress
2. Spustitelný kód aplikace Compress
3. Kolekce testovacích obrázků
4. Zbylé testovací tabulky
5. Elektronická verze bakalářské práce